1,108,801

PATENT SPECIFICATION

DRAWINGS ATTACHED

1,108,801

Date of Application and filing Complete Specification: 16 March, 1965.
No. 10969/65.
Application made in United States of America (No. 357372) on 6 April, 1964.
Complete Specification Published: 3 April, 1968.

Index at acceptance: —G4 A(2F10, 4R, 6M1, 8G, 12D, 12N, 15A2, 16D, 16G, 16H, 16J, 17P, 19)
Int. Cl.: —G 06 f 9/00

COMPLETE SPECIFICATION

Improvements in or relating to Electronic Data Processing Systems

We, INTERNATIONAL BUSINESS MACHINES CORPORATION, a Corporation organized and existing under the laws of the State of New York in the United States of America, of Armonk, New York 10504, United States of America (assignees of GENE MYRON AMDAHL et al) do hereby declare the invention for which we pray that a patent may be granted to us, and the method by which it is to be performed, to be particularly described in and by the following statement: —

The present invention relates to electronic data processing systems, and more specifically to stored program-controlled electronic data processing systems.

During the execution of the various operations performed in stored program-controlled electronic data processing systems, certain conditions may arise, which require an interruption of the normal problem program. Such conditions may occur in different parts of the system; they may be caused by an input/output device, for example, when an input or output operation is to be started or terminated by one of the I/O channels; they may arise in so-called external units, such as real time devices, the console, or a time used in the system; machine faults or program errors may, upon detection, also require an interruption. Under normal circumstances it is sufficient, upon occurrence of an interrupt condition, to interrupt the problem program being executed, to process an interrupt subroutine, and to proceed with the problem after completion of the sub-routine.

Heretofore such interrupts required extensive housekeeping operations in order to avoid losing the information developed during the execution of the problem program and to ensure that the problem program can resume at the point where it had been interrupted. Multiple simultaneously occurring requests for an interrupt were handled by performing the interrupt sub-routine of highest priority first while the machine parts requiring an interrupt of lower priority had to wait. This caused difficulties and sometimes even loss of information.

The object of the present invention is to provide a stored program controlled electronic data processing system having improved means for ensuring the correct resumption of a program which has been interrupted.

According to the present invention an electronic data processing system controlled by a program of stored instructions has means for interrupting the execution of a program on the occurrence of an event relating to the operation of the system and for replacing the program by a predetermined sub-routine of instructions which controls the response of the system to said event, and includes means for establishing, on the simultaneous occurrence of two or more such events, the relative priorities of the events, transfer means for replacing the program with the sub-routine associated with the event of highest priority, and storing means for storing information relating to the program and to the sub-routines associated with events of lower priority at predetermined locations in order that the other sub-routines may be executed and the interrupted program may be resumed when the interruption is concluded.

In order that the invention may be more fully understood reference will be made to the accompanying drawings, in which: —

Figure 1 is a simplified block diagram of an electronic data processing system employing an interrupt system in accordance with the invention;

Figure 2 shows the format of a Program Status Word (PSW);

Figure 3 is a diagram showing at what times requests for the different types of interrupt may occur in relation to the problem

[*Price 4s. 6d.*]

program, and at what times the interrupt sub-routines are started;

Figure 4 is a diagram showing in which sequence, upon occurrence of multiple interrupt requests, the PSW's related to the problem program and to the different interrupt sub-routines are up-dated and stored;

Figure 5 is a block diagram of the circuitry used to determine whether or not a requested interrupt is allowed according to the current PSW; and

Figure 6 is a block diagram of the circuitry determining the wired-in interrupt priority sequence.

Figure 1 shows a block diagram of a machine in which an interrupt system in accordance with the invention may be implemented.

In the diagram block 1 represents a core storage; it includes the main storage 2 used to store macro-instructions, other control words and data, all of which may be of different length, and a CPU storage section 3 which is called bump storage or storage bump and is used to perform register functions for the operations of the central processing unit (CPU) of the system. The eight bit storage locations are addressed by an address register 5; the stored eight bits, forming a byte, are read out in parallel into a data register 19. Data is read in under control of inhibit circuitry 4, associated with the storage, in accordance with the contents of the register 19.

Block 6 represents a read-only store containing a dictionary of micro-instructions. Micro-instructions are read out, from locations specified by a read-only store address register 7, into sense amplifier latches 8. After decoding, they are used to control the operation of the system, the control circuits being indicated by block 9. The control circuits include circuitry (10, 11) used to control the interrupt operations. This circuitry and its operations will be explained in more detail in connection with Figures 5 and 6. Blocks 12 indicate those parts of the system which may require an interrupt sub-routine.

Arithmetic and logic operations are performed in an arithmetic and logic unit (ALU) 18. The two operands upon which the arithmetic or logic operations have to be performed are entered into the ALU from A bus 13 and B bus 14 via buffer registers A(16) and B(17). Result signals are fed to Z bus 15 from where they are gated to one or more of the basic registers 19 to 25. These registers serve different purposes: as explained above, register 19 is used as main storage data register; register 20 serves as instruction counter, its contents being used to set the storage address register 5; register 21 serves as a preliminary address register, whose contents may be used to address storage 1 through address register 5, or read-only storage 6 through address register 7. The remaining registers 22 to 25 are used as temporary storage for operation codes, condition codes, length codes, status bits, partial addresses or other information relevant to the program currently processed. More registers than indicated may be provided. The register contents can be gated to the different buses and address registers as shown in the diagram; for example, data read out from main storage 2 into data register 19 may be routed through the ALU 18, via A bus 13 and A buffer 16, to any one of the other registers. In the system shown in Fig. 1, problem programs are processed under control of a sequence of macro-instructions, each of which in turn is executed under the direct control of a sequence of micro-instructions. The operation codes of the macro-instructions determine the operation to be performed, but instead of utilizing manifestations of the macro-instructions directly to control the circuits of the system, the operation code, stored in one of the basic registers, in connection with control circuitry 9, causes selection of a first macro-instruction from read-only storage 6. The address of the next micro-instruction in sequence is at least partly contained in the preceding micro-instruction, and is completed by taking the machine conditions and operation code of the macro-instruction into account. After execution of the sequence of micro-instructions associated with a macro-instruction, the next macro-instruction is fetched from the main storage location determined by the up-dated instruction counter 20, and the problem program processing is continued.

The system operates under the control of a monitor program which supervises all operations of the system. When several problem programs, each consisting of a sequence of macro-instructions, are to be processed, the monitor ensures their proper scheduling; it also uses sub-routine programs determining what action will be taken when an interrupt is requested owing to certain conditions occuring in the system.

During the execution of problem programs, the CPU operates under the control of a control word provided by the monitor program, called the program status word PSW, one PSW being available for the problem program and for each type of interrupt sub-routine. This PSW is a constantly up-dated record of the various conditions of the system, requiring a minimum of bit manifestations for selected operations of the various parts of the system. Among other information it contains the address of the next macro-instruction. For direct operation control the current or active PSW is kept in a fixed location in the CPU storage bump section

3 of storage 1; portions of it may also be held in registers, latches, or similar temporary storage devices in the CPU or the I/O channels. Inactive PSW's, relating to programs not currently being processed, are stored in fixed locations in main storage 2. Once a problem is interrupted, the corresponding PSW is stored as an "old" PSW, and a "new" PSW, related to the next sequence of operations, is fetched from main storage. These PSW's, at any time, contain all pertinent information necessary to allow a program to be interrupted and restarted as required.

Following is a more detailed description of the PSW control and of the operation of the interrupt system.

Fig. 2 shows a schematic diagram of the PSW (26) used in the system; it contains bit positions 0 to 63 grouped in fields 26A to 26K. An explanation of the information held in the different bit positions and of the control functions performed by that information is given below.

Field 26A—Bits 0 to 7—SYSTEM MASK

The bits of the system mask are used, upon occurrence of an I/O or external interrupt request, to determine whether or not the interrupt is allowed.

Bit positions correspond to the various interrupt sources as follows:

Bit	Interrupt Source
0	Multiplex Channel
1	Selector Channel 1
2	Selector Channel 2
3	Selector Channel 3
4	Selector Channel 4
5	Selector Channel 5
6	Selector Channel 6
7	External

Bits 0 to 6 being "1" will allow interrupts from the corresponding I/O channel to occur. Any of these bits being "0" will prevent an interrupt from the corresponding I/O channel. Likewise, bit 7 being "1" will allow an external interrupt to occur and bit 7 being "0" prevents it from occurring. The use of these interrupts is described in more detail later.

The system mask of the current PSW is stored in latches and in addition a reference image is maintained in a fixed location of the CPU storage bump.

Field 26 B—Bits 8 to 11—
PROTECTION KEY

This four bit code designates areas in storage which are accessible for instructions of current problem programs. A mismatch with the storage key, contained in a storage adress used to access storage, causes an interrupt.

The protection key of the current PSW is held in a register of the CPU. A reference image is also maintained in a fixed location of the CPU storage bump.

Field 26C—Bit 12—Not used.

Field 26D—Bit 13—
MACHINE CHECK MASK

This mask bit is used, upon occurrence of a request for a machine check interrupt indicating a machine fault, to determine whether or not the interrupt is allowed. If this bit is set to "1", a machine check will cause an interrupt to occur. If the bit is "0", the CPU will try to continue operating in spite of machine checks.

Field 26E—Bit 14—RUNNING/WAIT

If this bit is set to "0", the CPU executes instructions as normal. If this bit is "1", the CPU assumes a WAIT state. In this case, the CPU ceases to execute instructions and waits for an interrupt to occur before executing further instructions.

Field 26F—Bit 15—
MONITOR/PROBLEM

If this bit is set to "1", the CPU is in the problem program state. If this bit is "0", the CPU is in the monitor state in which only priviledged functions can be executed; other functions will develop an error indication.

The current PSW holds an image of bits 13, 14, and 15 in the CPU bump. Bits 13 and 14 are also held in latches.

Field 26G—Bits 16 to 31—
INTERRUPT CODE

This 16 bit code is used to describe the cause of the interrupt; it determines, for the different types of interrupt, the unit requesting the interrupt as well as the reason for or the condition requiring an interrupt.

There is no interrupt code associated with the current PSW. This information is pertinent only when storing the old PSW in case of an interrupt.

Field 26H—Bits 32 and 33—
INSTRUCTION LENGTH CODE

Macro-instructions of different length are used in the system. The eight bit bytes, forming an instruction, are stored in successive storage locations. The instruction length code is used by the monitor program in the case of a program check or a machine check to determine the first byte address of the instruction that caused the interrupt. Bits 32 and 33 describe the length of the instruction just completed or being executed. They are loaded by each subsequent macro-instruction, bit positions 0 and 1 of which specify the length of the instruction, so that the storage address of the start of the instruction can be regained.

Field 26I—Bits 34 and 35—
CONDITION CODE

These bits describe one of several conditions that may have resulted from the execution of an instruction. In the current PSW these bits are always held in a fixed location in the CPU storage bump in a decoded 1-out-of-4 form as follows:

Bits 34 and 35	4 Bit Code of Current PSW
00	1000
01	0100
01	0010
11	0001

Each time an instruction is performed which affects the condition register used in the system, this four bit code is changed to indicate the state of the condition register. The code then remains unchanged until another instruction which affects the condition register is executed. The use of the condition code is explained more fully in our copending Application 10973/65 (Serial No. 1108802).

Field 26J—Bits 36 to 39—
PROGRAM MASK

The program mask is used, upon occurrence of a request for a program check interrupt indicating a program error, to determine whether or not the interrupt is allowed.

The program mask bits correspond to four permissible interrupt conditions which may occur within the CPU. Each time one of these conditions occurs, the program mask is interrogated, and if the corresponding bit is a "1" a program check interrupt is taken. If the bit is "0", the program proceeds.

The program mask of the current PSW is held in fixed location of the CPU storage bump.

Field 26K—Bits 40 to 63—
INSTRUCTION ADDRESS

These bits define the address of the next macro-instruction to be executed in the current program.

After execution of a macro-instruction, the address of the next macro-instruction in the current program is contained in the instruction counter as well as in the associated bit positions of the current PSW. For each macro-instruction executed the stored address is incremented in accordance with the instruction length code contained in the instruction; the address is replaced when the normal instruction sequence is modified by a branch instruction.

The interrupt system employed in the data processing system permits the CPU to interrupt a current program and to execute an interrupt sub-routine as a result of conditions which arise external to the system, in I/O units, or in the CPU itself. An interruption includes the storing of the current PSW as an "old" PSW, and the fetching of a "new" PSW. Processing resumes in the state indicated by the new PSW.

There are five types of interrupt which may occur in the system:

1. MACHINE CHECK

A machine check interrupt occurs as a result of a machine malfunction.

2. PROGRAM CHECK

A program check interrupt occurs as a result of finding a fault in the program such as invalid operation code, improper specification, or overflow conditions.

3. INPUT/OUTPUT

An I/O interrupt occurs as a result of an I/O device reaching a point in its operation where intervention by the monitor program is required.

4. EXTERNAL

The external interrupt provides a means by which the CPU responds to signals from the timer, from the console interrupt key, or from external real time data channels.

5. MONITOR CALL

This interrupt occurs as a result of the monitor call instruction. One of the main purposes of this interrupt is to switch the CPU from the problem state to the monitor state, this being required when a PSW has to be replaced or altered.

In order for a machine check interrupt, a program check interrupt, an I/O interrupt, or an external interrupt to occur, the mask bits in the current PSW, corresponding to the source or type of interrupt, must be a "1". Should this bit be a "0", the interrupt cannot occur until that bit is made "1".

A monitor call interrupt is initiated by a macro-instruction, thus not requiring a mask bit in the PSW.

Fig. 3 shows a diagram indicating at what time, with respect to the problem program, requests for the different types of interrupt may occur and at what time the interrupts are taken. As explained above, an operation, defined by a macro-instruction, is executed under direct control of a sequence of micro-instructions 1 to n determined by that macro-instruction (columns 30A, 30B).

Arrows in the column "time at which request for interrupt may occur" (30C) indicate, for each type of interrupt, at what time, in relation to the micro-instruction sequence, the request may occur.

Requests for machine check (MACH CHECK), program check (PROG CHECK), Input/Output (I/O), and external (EXT) interrupts may occur at any time, as indicated

by the upper arrow and the large bracket. The monitor call (MON CALL) interrupt is initiated by a macro-instruction provided by the programmer and can therefore occur only after completion of the micro-instruction sequence, as indicated by the lower arrow marked MON CALL.

Arrows in the column "time interrupt is taken" (30D) indicate, for each type of interrupt, at what time, in relation to the micro-instruction sequence, the interrupt sub-routine for an allowed interrupt is started if it had been requested.

As indicated, MACH CHECK and PROG CHECK interrupts are processed immediately. The current micro-instruction sequence is terminated; its completion could lead to erroneous results, as these interrupts indicate machine faults or improper programs.

I/O and EXT interrupts are processed after completion of the current micro-instruction sequence. The last micro-instruction in sequence is always used to interrogate whether or not an I/O or an EXT interrupt had been requested during the execution of the micro-instruction sequence.

A MON CALL interrupt is processed after termination of a micro-instruction sequence as explained above.

Through the use of PSW's, an interrupt system is provided in which simultaneously occurring interrupts may be handled on the basis of priority without losing the ability to handle interrupts of a lower priority. The system is capable of preserving all information relating to multiple interrupts by successively storing the old problem program PSW, fetching the new PSW relating to a first (lowest priority) interrupt, storing this first new PSW without instruction execution, fetching the next new PSW relating to the next interrupt in priority sequence, etc. Instruction execution is resumed using the PSW which was fetched last (highest priority), the order of execution of the interrupt sub-routines being the reverse of the order in which the PSW's were fetched.

In main storage there are ten sixty-four bit storage locations reserved for storing PSW's. The PSW's for which these locations are provided are named as follows:

I. Old External PSW
II. New External PSW
III. Old I/O PSW
IV. New I/O PSW
V. Old Machine Check PS W
VI. New Machine Check PSW
VII. Old Program Check PSW
VIII. New Program Check PSW
IX. Old Monitor Call PSW
X. New Monitor Call PSW

When an interrupt occurs, the current updated problem program PSW is stored as an old PSW in the appropriate location. For example, when a PROG CHECK occurs, the up-dated old PSW is stored in location VII, "Old Program Check PSW". Most bit positions of the PSW are stored exactly as they were held in the CPU storage bump or in the temporary CPU storage devices. However, in the case of the condition register, a decoding must be done as explained previously.

When storing an old PSW, an interrupt code is always stored in bit positions 16 to 31. This code identifies the cause of the interrupt as follows:

1. For an EXT interrupt, the code identifies which one of the several possible items caused the interrupt.
2. For a MON CALL interrupt, the code contains an eight bit message which is decoded by the monitor program to determine the reason for the monitor call instruction.
3. For a PROG CHECK interrupt, the code identifies which one of the possible items, such as invalid operation, improper specification or overflow, caused the interrupt.
4. For a MACH CHECK interrupt, the code is always set to 0.
5. For an I/O interrupt, the code identifies the chanel and the unit requesting the interrupt.

As soon as the old PSW is completely stored, the corresponding new PSW is loaded into the CPU storage bump location provided for the current PSW, and into temporary CPU storage devices. For example, when an old program PSW is stored in location VII, "Old program check PSW", the new PSW for the control of the program check interrupt sub-routine is loaded from location VIII "New Program Check PSW". When loading a new PSW, the interrupt code is not pertinent and is ignored, while the condition code is decoded from 2 bits to 4 bits as explained previously. Then, the new PSW contains the start address and starting conditions for the monitor program which is to handle the interrupt sub-routine, and the CPU resumes execution of instructions until the interrupt sub-routine is completed. At this time, the PSW controlling the interrupt sub-routine is led back into its associated location in storage. The old PSW, used to control the interrupted problem program, is then fetched from storage and CPU will again start to execute instructions, starting with the one it was about to perform when the interrupt occurred.

Fig. 4 is a diagram showing how the PSW's are used when two requests for allowed interrupts, external and I/O, occur simultaneously. The operation which determines whether or not an interrupt is allowed is explained below in connection with Fig. 5.

In Fig. 4 column 40A contains figures

relating to the different operational steps required for the assumed example. Columns 40B and 40C indicate which program or sub-routine is processed at any time and which PSW is held as current or active PSW in the CPU storage bump, the different programs and sub-routines, problem program, external interrupt, and I/O interrupt being represented by hatchings in accordance with those shown in blocks 41, 42, and 43. Columns 40D to 40H represent pairs of fixed main storage locations used to store inactive PSW's, a pair of locations I to X being provided for the "old" and the "new" PSW's for each type of interrupt, i.e. I and II for EXT interrupts, III and IV for I/O interrupts, etc. Storing a PSW in the "old" location of a pair is automatically followed by the read-out of the associated "new" location.

Following is a detailed explanation of the operation for each of the steps 1 to 12, illustrated in Figure 4:

Step 1:
Processing of a problem program under control of a problem program PSW stored in the CPU storage bump. The "old" PSW storage locations in main storage (odd numbers I, III, etc) are empty, while the "new" PSW locations (even numbers II, IV, etc.) contain the PSW's corresponding to the different interrupt sub-routines.

Step 2:
During execution of the problem program, requests for two allowed interrupts of different priority occur simultaneously: a request for an EXT interrupt (lower priority) and a request for an I/O interrupt (higher priority). In accordance with an operation described below in detail with reference to Fig. 6, the problem program PSW is stored as an "old" PSW in location I, this being the location corresponding to the lowest priority interrupt requested.

Step 3:
The "new" external interrupt PSW is fetched from location II and stored in the CPU storage bump. As a second interrupt request is waiting, processing of the external interrupt sub-routine is not started.

Step 4:
The EXT interrupt PSW is stored as an "old" PSW in location III, this being the location corresponding to the requested interrupt which is the next in priority sequence.

Step 5:
The "new" I/O interrupt PSW is fetched from location IV and stored in the CPU storage bump. As no further interrupt is requested, processing of the I/O interrupt sub-routine is started.

Step 6:
Processing of the I/O interrupt sub-routine under control of the I/O interrupt PSW stored in the CPU storage bump.

Step 7:
After completion of the I/O interrupt sub-routine, the I/O PSW is fed back into its associated storage location IV.

Step 8:
The EXT interrupt PSW is automatically fetched from location III and stored in the CPU storage bump. Processing of the EXT interrupt sub-routine is started.

Step 9:
Processing of the EXT interrupt sub-routine under control of the EXT interrupt PSW stored in the CPU storage bump.

Step 10:
After completion of the EXT interrupt sub-routine, the EXT PSW is fed back into its associated storage location II.

Step 11:
The problem program PSW is fetched from location I and stored in the CPU storage bump. Processing of the problem program is resumed at the point where it had been interrupted by step 2.

Step 12:
Processing of the problem program under control of the problem program PSW stored in the CPU storage bump.

The handling of the different PSW's has been explained using the example where two interrupt requests occur simultaneously. Cases of single interrupts or of interrupts of an interrupt sub-routine require no further explanation as they are performed in a corresponding manner. The same applies for types of interrupt other than those chosen for the example.

Fig. 5 shows a block diagram of the arrangement 11 used for the masking operation which determines whether or not a requested interrupt is, at the time of its occurrence, allowed. The circuitry as shown is restricted to the handling of I/O and EXT interrupts requested by I/O channels 120 to 126, or by external sources 127.

As explained above, the bits of the system mask field of the current PSW 26 are used to determine whether or not a requested I/O or EXT interrupt is allowed. The bit corresponding to the interrupt source (bit 0 for the multiplex channel, bits 1 to 6 for the six selector channels 1 to 6, and bit 7 for the external interrupt sources) being a "1" specifies the interrupt is to be allowed; a "0" would indicate that the interrupt is to be rejected at least temporarily. The

complete current PSW 26 is stored in the CPU storage bump 3. The system mask information is also contained in associated latches 500 to 507, a latch being set to its ON state if the corresponding mask bit is a "1", and to its OFF state if it is a "0". For each system mask bit, respectively for each of the interrupt sources 120 to 127, one of the two-way AND circuits 530 to 537 is provided, the input lines of the AND circuits being connected via lines 510 to 517 to the corresponding latches, and via lines 520 to 527 to the corresponding interrupt source. A latch being in its ON state provides a signal for its associated AND circuit enabling an interrupt request signal received from the associated interrupt source to appear at the output of the AND circuit. These output signals appearing on one of the output lines 540 to 547 are used to indicate a request for an allowed interrupt. The arrangement shown in Fig. 5 also provides signals "interrupt requested" (INT REQU) and "external interrupt requested" (EXT REQU) on lines 57 and 58 respectively, indicating that an interrupt is requested by any one of the interrupt sources or external interrupt sources, respectively. The signal INT REQU is obtained by simple OR circuitry 56 as shown.

As a plurality of I/O channels 120 to 126 is employed, which may request an interrupt at the same time, a priority sequence among these channels must be established unless all channels except one are masked off by the current PSW. This priority circuitry, however, is not shown, as it is not an essential part of the invention.

Fig. 6 shows a block diagram of control circuitry 10 used to handle the different PSW's in the case when single or multiple requests for allowed interrupts occur. This circuitry determines the wired-in priority sequence in which interrupt sub-routines are taken.

Block 60 represents the circuits which, under direct control of a sequence of micro-instructions obtained from read-only storage, control the execution of the macro-instructions of the program being processed. These operations are initiated at the beginning of a problem program by a signal on line 61, or by a signal occurring on line 62 after completion of the execution of a macro-instruction if no interrupt request is waiting, and at the beginning of an interrupt sub-routine, the operations include the up-dating of current PSW's.

Upon receipt of request signals for allowed MACH CHECK or PROG CHECK interrupts on lines 63 or 64 respectively, control 60 causes immediate termination of the micro-instruction sequence being executed, and provides signals on output lines 65 or 66 respecively which, gated to blocks 80D or 80E, initiate corresponding interrupt operations. When control 60 operates upon a monitor call instruction provided by the programmer, a signal is developed on output line 67 which, gated to block 80C, initiates the required interrupt operation.

The last micro-instruction in a sequence corresponding to a macro-instruction always causes a signal on line 68, which is used to interrogate whether or not a request for an I/O or an EXT interrupt had occurred during execution of the micro-instruction sequence.

Blocks 70 "INTERRUPT?" and 72 "EXTERNAL?" consist of simple logic circuitry performing the following function on interrogation signals received on input lines 68 or 71, respectively: when an interrupt request signal is present on input lines 57 or 58, a signal is developed on YES output line 71 or 73, respectively; in the absence of a request signal, a signal is developed on the NO output line 62 or 74, respectively.

Blocks 80A to 80E represent circuits, one for each of the interrupt types EXT, I/O, MON CALL, PROG CHECK, and MACH CHECK, which, under micro-instruction control, initiate the following operations: updating of the current PSW with information indicating the reason and the source of the interrupt; transferring the current PSW from the CPU storage bump to one of the old PSW storage locations, the location being determined by the type of interrupt: Old External PSW for block 80A, Old I/O PSW for block 80B, Old Monitor Call PSW for block 80D, Old Program Check PSW for block 80D, and Old Machine Check PSW for block 80E; loading of a new PSW, from that main storage location associated with the location in which the old PSW had been stored, into the CPU storage bump location provided for the current PSW. After completion of these operations, a signal is developed on output lines 68 A, B, C, D, E, resp.

The operation of the control circuitry shown in Fig. 6 is now explained by describing three examples:

1. No interrupt requested
2. Single interrupt requested
3. Multiple simultaneously occurring interrupts requested.

EXAMPLE 1

It is assumed that, during the execution of a macro-instruction performed under control of its corresponding micro-instruction sequence, a request for an allowed interrupt did not occur.

As no MACH CHECK or PROG CHECK interrupts are requested which would cause immediate termination of the micro-instruction sequence, the last micro-instruction in sequence is used to interrogate whether or not a request for an I/O or an EXT interrupt is waiting. The interrogation signal on line 68 is gated to the circuitry indicated

by block 70. This circuitry, also receiving a signal INT REQU on line 57 if a request for an EXT or an I/O interrupt is waiting, performs simple logic functions to deliver either one of the output signals YES or NO on lines 71 or 62. As it is assumed that no interrupt had been requested, line 57 is down and a signal appears on the NO output line 62. This signal, fed to control circuitry 60, is used to initiate processing of the next macro-instruction of the current program.

EXAMPLE 2

It is assumed that during execution of a macro-instruction belonging to a problem program, an I/O interrupt is requested by selector channel 1, and that the system mask bit of the current PSW, corresponding to selector channel 1, is a "1", thus specifying that it is an allowed interrupt.

As no MACH CHECK or PROG CHECK interrupts are requested which would cause immediate termination of the micro-instruction sequence, the last micro-instruction in sequence is used to interrogate whether or not a request for an I/O or an EXT interrupt is waiting. The interrogation signal on line 68 is gated to the circuitry indicated by block 70. As it is assumed that a request for an allowed interrupt had occurred, line 57 is up and a signal is developed on the YES output line 71 and fed to the circuitry indicated by block 72. This circuitry, also receiving a signal EXT REQU on line 58 when a request for an EXT interrupt is waiting, performs the same functions as block 70 to deliver either one of the output signals YES or NO on lines 73 or 74. As it is assumed that only an I/O interrupt had been requested, line 58 is down and a signal is developed on NO output line 74 which, for the arrangement shown in Fig. 6, indicates that an I/O interrupt had been requested. This becomes obvious upon the following consideration: the signal on line 57 indicates that a request for any type of interrogated interrupt is waiting; and as in the present arrangement two different types of interrupt are interrogated, and the absence of the signal on line 58 had indicated that the request was not for an EXT interrupt, the request necessarily must be an I/O request.

The interrogation of the different types of interrupt request is performed in a sequence with increasing priority. In the arrangement shown in Fig. 6 only two priority levels are taken into account, but, without changing the basic concept involved, a more detailed priority scheme, e.g. for different priorities among the external or the I/O interrupt sources, could be used by adding further interrogation circuitry stages.

Returning now to the example, the NO signal on line 74 is fed to control circuitry 80B which controls the updating of the current PSW with information relevant to the interrupt, and the transfer of the old PSW associated with the current program from the CPU storage bump, to main storage location "Old I/O PSW". As the interrogation for EXT and I/O interrupts takes place at the end of the execution of a macro-instruction, the current PSW at this time contains all the information required to resume execution of the program after completion of an interrupt sub-routine. No further housekeeping operations are required besides storing the current PSW as an old PSW. Then, still under control of circuitry 80B, the new PSW, required for the I/O interrupt sub-routine, is fetched from main storage location "New I/O PSW" and loaded into the CPU storage bump location provided for the current PSW.

After completion of the PSW transfer operation, the output signal on line 68B is gated to block 70 and the interrogating procedure, as explained above, is repeated. As it was assumed that no further interrupts had been requested, a signal on NO output line 62 is developed and used to initiate processing of the I/O interrupt sub-routine which is performed under control of circuitry 60.

As soon as this sub-routine is completed and assuming that no further interrupt requests have occurred, the current PSW, which is the I/O interupt PSW, is stored in main storage location "New I/O PSW", and the PSW containing the necessary information to resume the problem program at the point where it was interrupted is fetched from location "Old I/O PSW" and loaded into the CPU storage bump. Execution of the problem program is then continued.

EXAMPLE 3

It is assumed that, during execution of a macro-instruction belonging to a problem program, two interrupts of different priority, an I/O interrupt and an EXT interrupt, are requested and found to be allowed.

As no MACH CHECK or PROG CHECK interrupts are requested which would cause immediate termination of the micro-instruction sequence, the last micro-instruction in sequence is used to interrogate whether or not a request for an I/O or an EXT interrupt is waiting. The interrogation signal on line 68 is gated to the circuitry indicated by block 70. In accordance with the operation described under Example 2, block 70 develops a YES output signal on line 71, and block 72, both of its inputs being up, also delivers a YES signal on line 73. The latter signal is fed to control circuitry 80A, which initiates storing of the current PSW in storage location "Old EXT PSW" and subsequently fetching of the new PSW, required for the EXT interrupt sub-routine, from main storage location "New EXT PSW". The new PSW is

loaded into the CPU storage bump location provided for the current PSW.

After completion of these transfers, the signal developed on output line 68A is gated to block 70 and the interrogation procedure is repeated. As a second interrupt request is waiting, line 57 is still up and a signal is developed on YES output line 71. As the EXT interrupt request has been taken care of by fetching the PSW for the corresponding sub-routine, line 58 is down and block 72 develops a NO output signal on line 74. This indicates, as explained above, that an I/O interrupt is requested, and a signal is fed to block 80B which in turn initiates storing of the "old" PSW in predetermined main storage location "Old I/O PSW". It is noted that this "old" PSW is the control word for the EXT interrupt sub-routine, processing of which has not been started. This PSW contains all the information required to initiate the EXT interrupt sub-routine once it is again loaded into the CPU storage bump. As will be explained below, this will occur after completion of the I/O interrupt sub-routine which has higher priority.

After storing the old PSW, the new PSW required for the I/O interrupt sub-routine is fetched from main storage location "New I/O PSW" and loaded into the CPU storage bump location provided for the current PSW.

After completion of these operations the signal developed on line 68B is gated to block 70 and the interrogation procedure is repeated again. As it was assumed that no further interrupt had been requested, a signal on the NO output line 62 is developed and used to initiate processing of the ILO interrupt sub-routine under control of circuitry 60.

As soon as the I/O sub-routine is completed and assuming that no further interrupt requests have occurred, the current PSW, which is the I/O interrupt PSW, is stored in main storage location "New I/O PSW", the PSW for the EXT interrupt sub-routine is fetched from location "Old I/O PSW" and loaded into the CPU storage bump, and execution of the second interrupt sub-routine is started.

When this sub-routine is completed, and again assuming that no further interrupt requests have occurred in the meantime, the current PSW, which is in the EXT interrupt PSW, is stored in main storage location "New EXT PSW", the PSW containing the necessary information to resume the interrupted program is fetched from location "Old EXT PSW" and loaded into the CPU storage bump, and execution of the interrupted problem program is continued.

It has been explained that, during the operation of handling multiple interrupt requests described above, the PSW's are stored in order of increasing priority, the PSW of the lowest priority interrupt being stored first, while processing of the interrupt sub-routines is performed in the reverse sequence, the highest priority interrupt sub-routine being executed first.

The interrupt system in accordance with the invention is also capable of handling interrupt requests occurring during the execution of an interrupt sub-routine. If such a requested interrupt is allowed, the system is capable of interrupting the current interrupt sub-routine.

From the foregoing it becomes apparent that, once a monitor program has started executing an interrupt sub-routine, it must not be interrupted by the same type of interrupt until the program is completed. If this were to happen, all reference to the original program, operating at the time the first interrupt occurred, would be lost as the PSW controlling the first interrupt sub-routine would be stored in the Old PSW storage location which already contains the PSW of the original program. This situation is prevented by proper use of the PSW masks as follows: assuming that the two interrupts are of the I/O type, the system mask of the "new" I/O PSW, controlling the first interrupt sub-routine, must specify all other I/O interrupts to be masked off. In this manner, no further allowed I/O interrupts can occur at least until the present one has been completed.

MACH CHECK and PROG CHECK interrupts are processed immediately after their occurrence as explained above. These interrupts are also handled by storing the old PSW of the interrupted program and fetching the new PSW used to control the interrupt sub-routine. The same applies for MON CALL interrupts, this process being different in that it is initiated by a macro-instruction provided by the programmer.

The priority sequence, resulting from the arrangement shown in Fig. 6, can easily be altered by proper setting of the system masks of the PSW's. If, for any reason, an interrupt sub-routine of "lower" priority, according to the wired-in sequence, is to be executed before a simultaneously requested "higher" priority interrupt sub-routine is processed, those mask bits of the lower priority PSW which correspond to the higher priority interrupt sources can be set to "0", thereby specifying that these interrupts are not allowed. This provides for a very flexible interrupt system, capable of handling interrupts of different priority in a sequence determined by the programmer or by the program itself.

WHAT WE CLAIM IS:—

1. An electrode data processing system controlled by a program of stored instructions, having means for interrupting the execution of a program on the occurrence of an event

relating to the operation of the system and for replacing the program by a predetermined sub-routine of instructions which controls the response of the system to said event, and including means for establishing, on the simultaneous occurrence of two or more such events, the relative priorities of the events, transfer means for replacing the program with the sub-routine associated with the event of highest priority, and storing means for storing information relating to the program and to the sub-routines associated with events of lower priority at predetermined locations in order that the other sub-routines may be executed and the interrupted program may be resumed when the interruption is concluded.

2. An electronic data processing system as claimed in claim 1, in which the interrupt means may operate to interrupt the sub-routine replacing the interrupted program and in which, in such a case, said transfer means operates to replace said interrupted sub-routine by a sub-routine determining the response of the system to the event causing the interruption.

3. An electronic data processing system as claimed in claim 1 or claim 2, in which the said information comprises control words, one of which is associated with the said program and the remaining control words are associated respectively with different ones of said sub-routines associated with events causing the interruption, each control word being used to control the execution of the associated program or sub-routine.

4. An electronic data processing system as claimed in claim 3, comprising a register for storing the control word associated with a program or sub-routine being executed.

5. An electronic data processing system as claimed in claim 3 or claim 4, in which said storing means stores the control word associated with the interrupted program or sub-routine at a storage location, the address of which is determined by the nature of the event causing the interruption.

6. An electronic data processing system as claimed in claim 3, claim 4 or claim 5, in which a control word contains information by means of which interruptions of the associated program or sub-routine may be selectively prevented.

7. An electronic data processing system as claimed in any of claims 3 to 6, in which a control word contains information relating to machine conditions arising due to machine operations.

8. An electronic data processing system as claimed in any one of claims 3 to 7, in which a control word contains the storage address of an instruction of the associated program or sub-routine, said instruction being either the instruction currently being executed, or the instruction which was being executed when the associated program or sub-routine was interrupted.

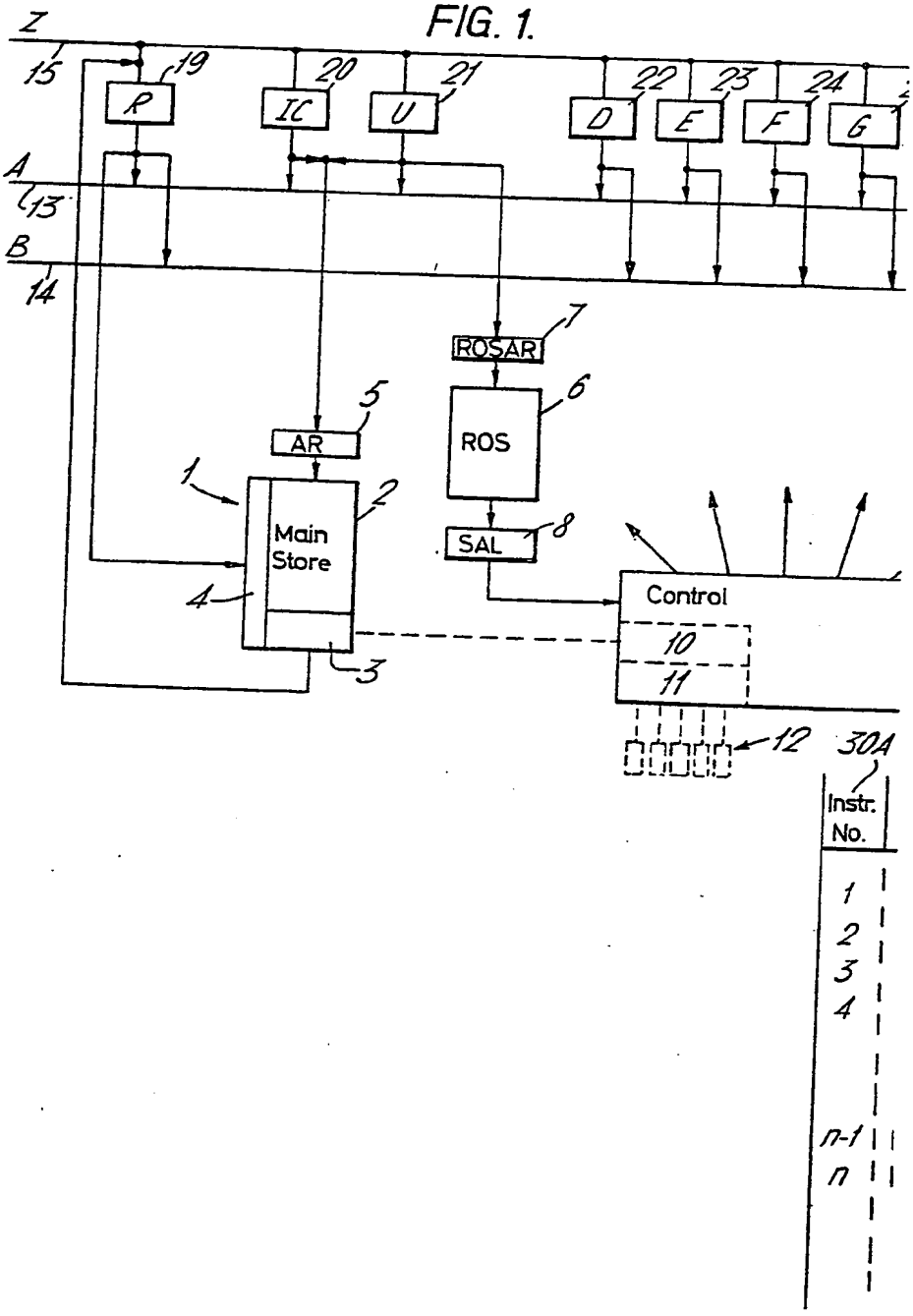
9. An electronic data processing system as claimed in any one of claims 3 to 8, comprising means for storing in the control word associated with the interrupted program or sub-routine, when the execution of a program or sub-routine is interrupted, information relating to the nature of the interruption.

10. An electronic data processing system as hereinbefore described with reference to the accompanying drawings.

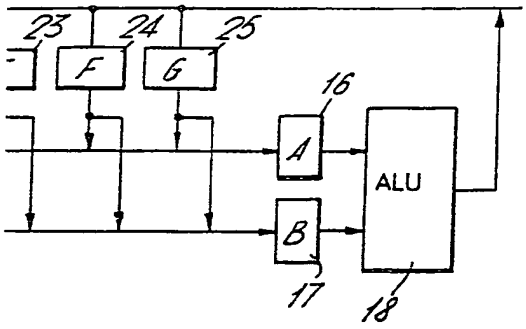
M. J. W. ATCHLEY,
Chartered Patent Agent,
Agent for the Applicants.

Printed for Her Majesty's Stationery Office by the Courier Press, Leamington Spa, 1968.
Published by the Patent Office, 25, Southampton Buildings, London, W.C.2, from which copies may be obtained.

FIG. 1.


I
15
19
20
21
22
23
24
R
IC
U
D
E
F
G
A
13
B
14
ROSAR
7
AR
5
ROS
6
1
2
Main Store
4
3
SAL
8
Control
10
11
12
30A
Instr. No.
1
2
3
4
n-1
n

1108801 COMPLETE SPECIFICATION
5 SHEETS This drawing is a reproduction of the Original on a reduced scale
Sheet 1



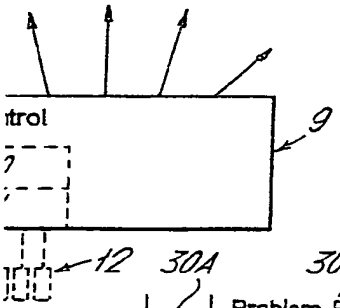
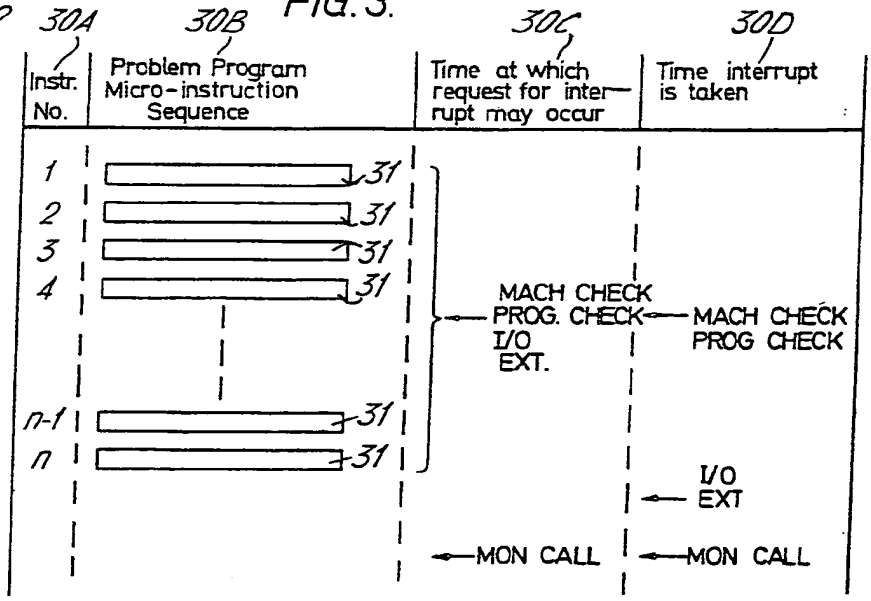


FIG. 3.



1108801 COMPLETE SPECIFICATION
5 SHEETS This drawing is a reproduction of the Original on a reduced scale
Sheet 1

FIG. 1.

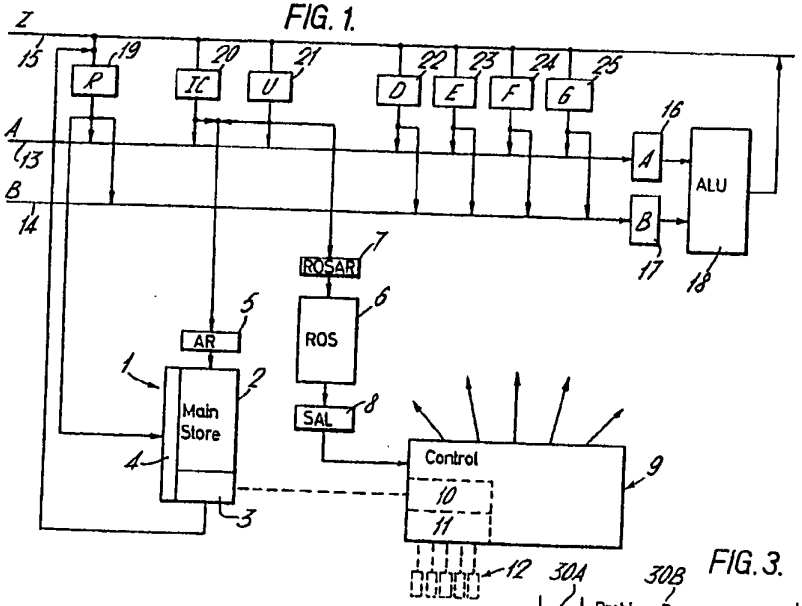


FIG. 3.

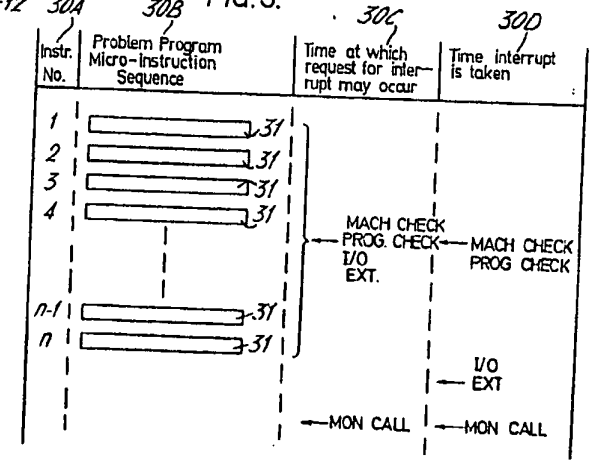
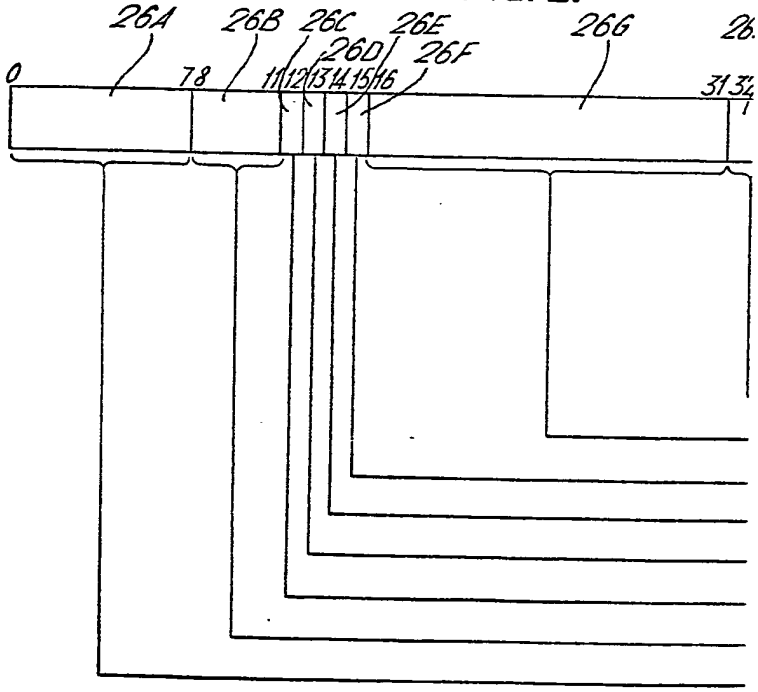
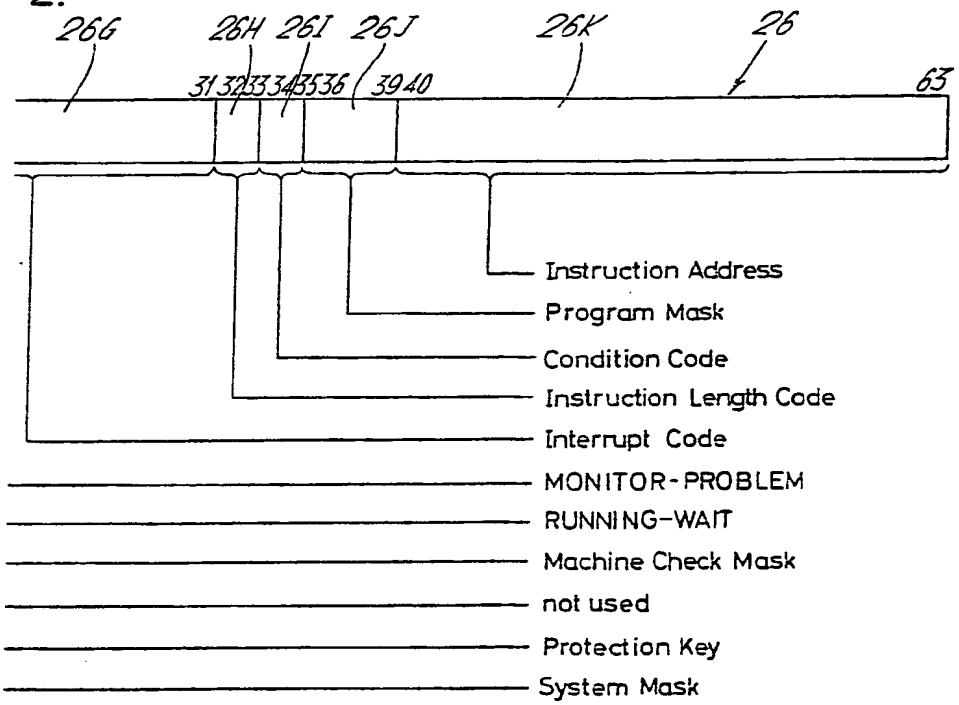


FIG. 2.



1108801 COMPLETE SPECIFICATION
5 SHEETS *This drawing is a reproduction of the Original on a reduced scale*
Sheet 2

2.



1108801 COMPLETE SPECIFICATION
5 SHEETS This drawing is a reproduction of the Original on a reduced scale
Sheet 2

FIG. 2.

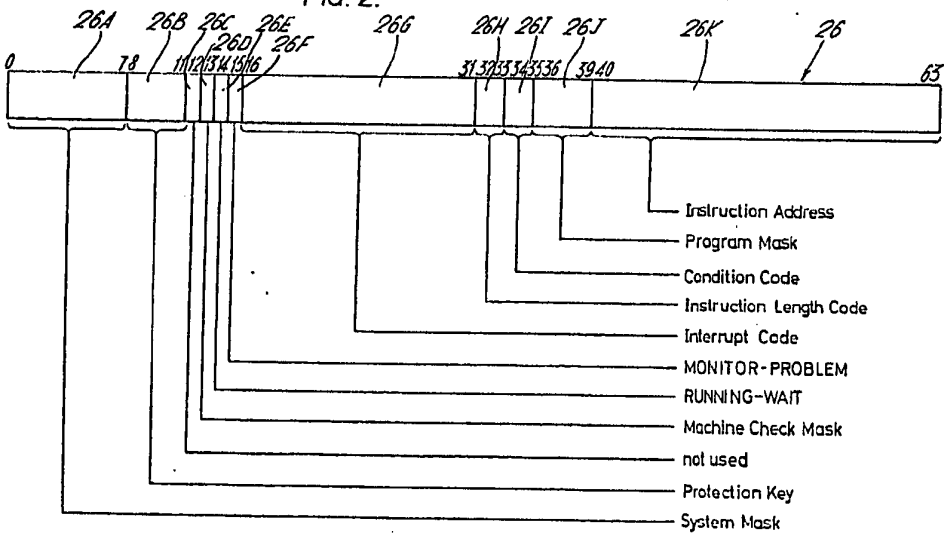
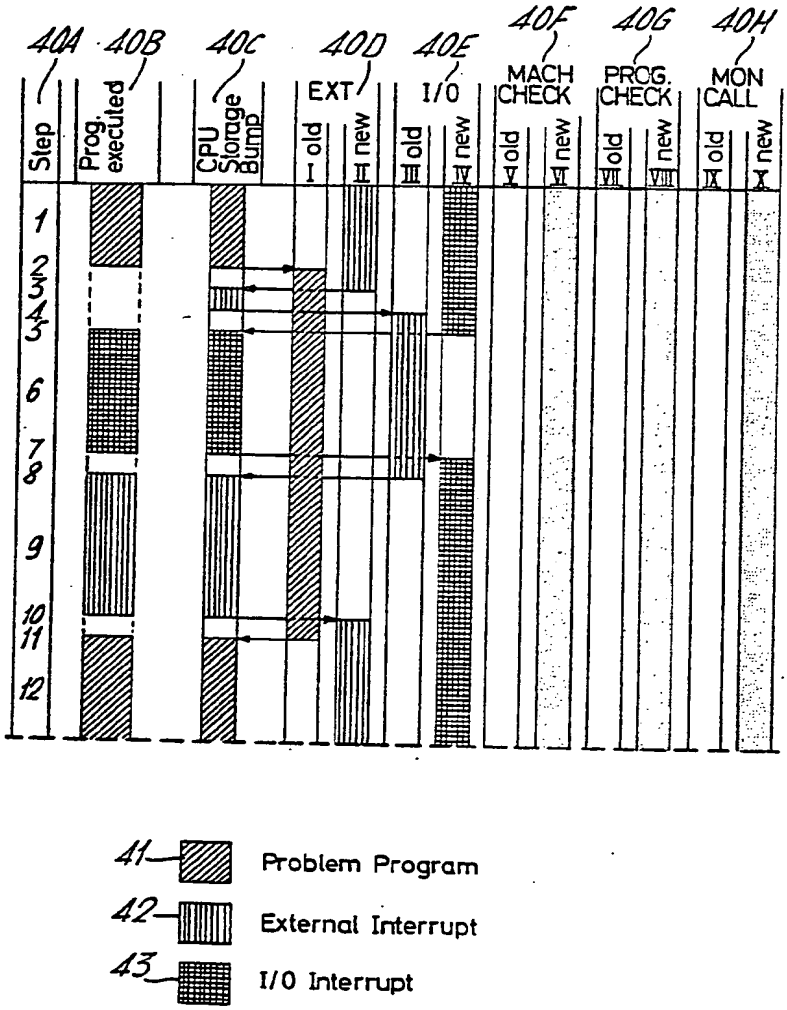
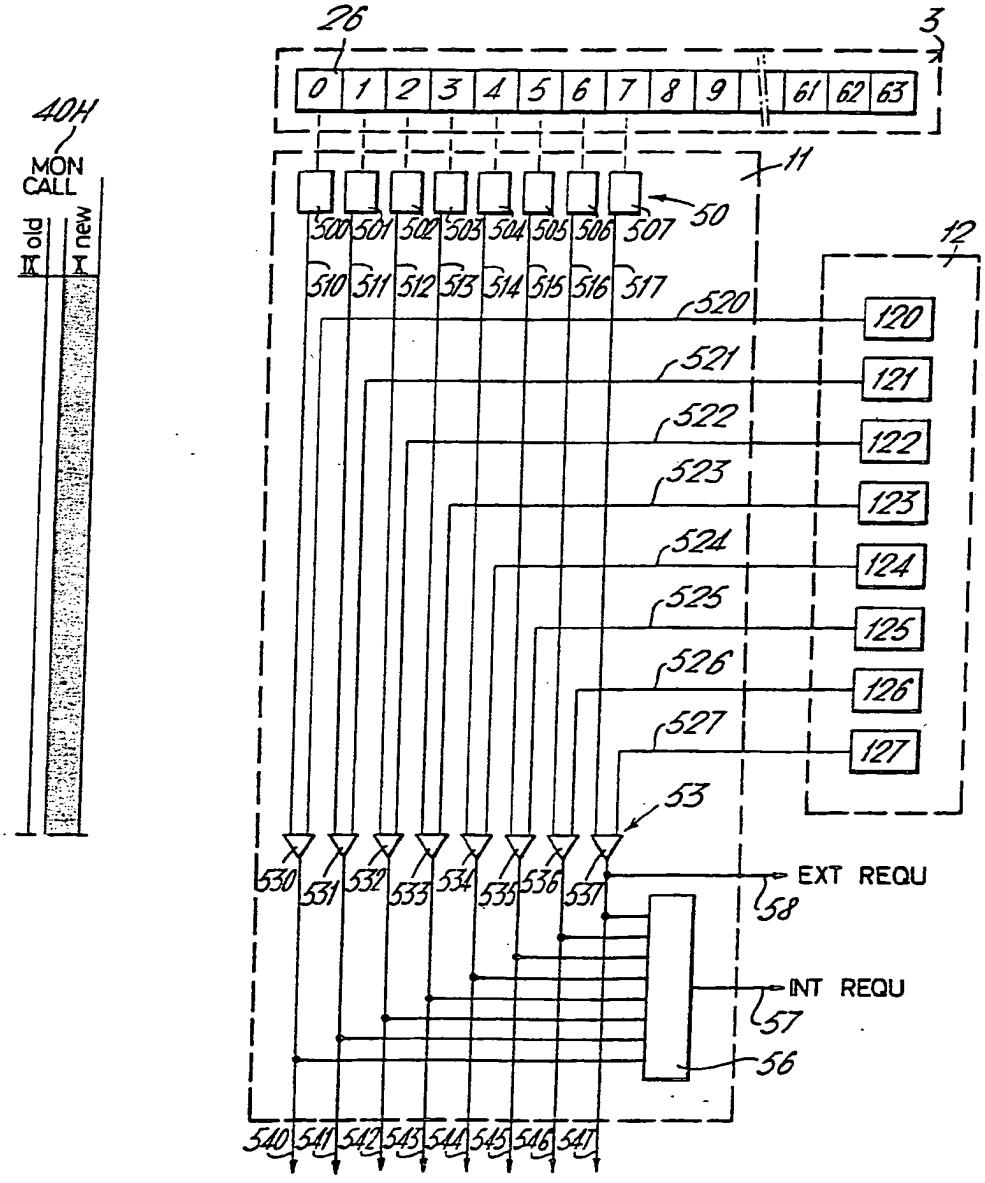


FIG. 4.


40A
40B
40C
40D
40E
40F
40G
40H
Step
Prog. executed
CPU Storage Bump
EXT
I/O
MACH CHECK
PROG. CHECK
MON CALL
old
new
I
II
III
IV
V
VI
VII
VIII
IX
X
1
2
3
4
5
6
7
8
9
10
11
12
41
Problem Program
42
External Interrupt
43
I/O Interrupt

1108801 COMPLETE SPECIFICATION
5 SHEETS This drawing is a reproduction of the Original on a reduced scale
Sheets 3 & 4

FIG. 5.



1108801 COMPLETE SPECIFICATION
5 SHEETS This drawing is a reproduction of the Original on a reduced scale
Sheets 3 & 4

FIG. 4.

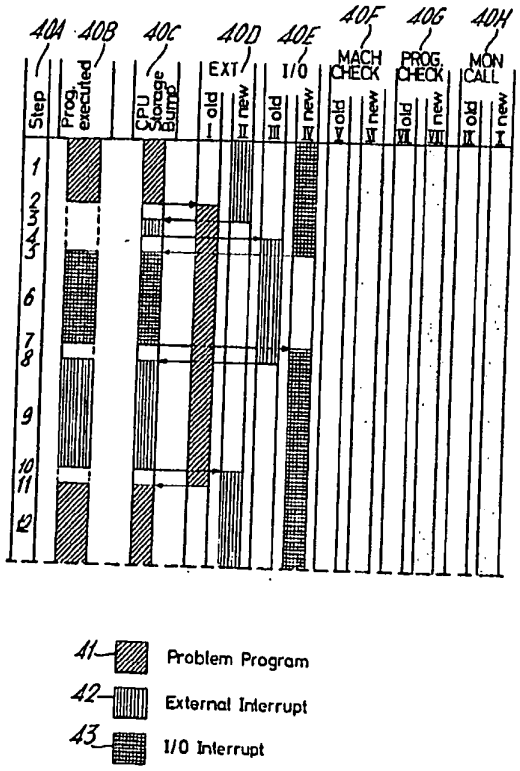
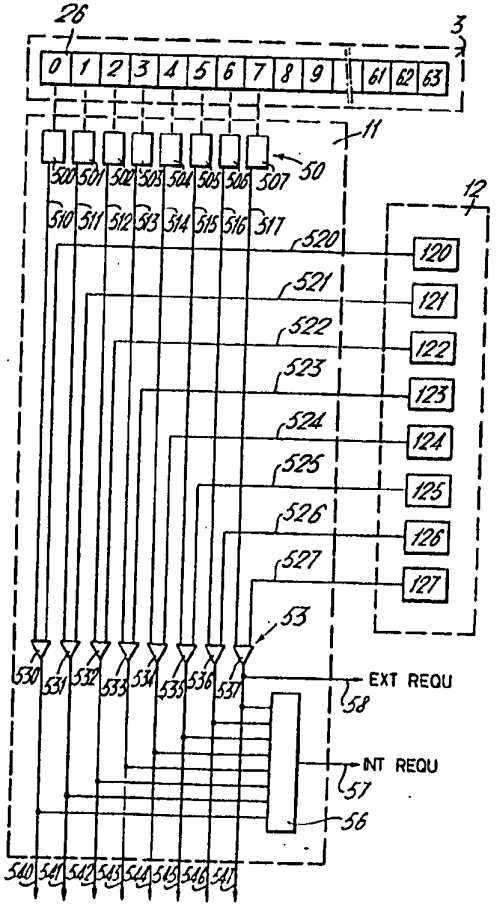
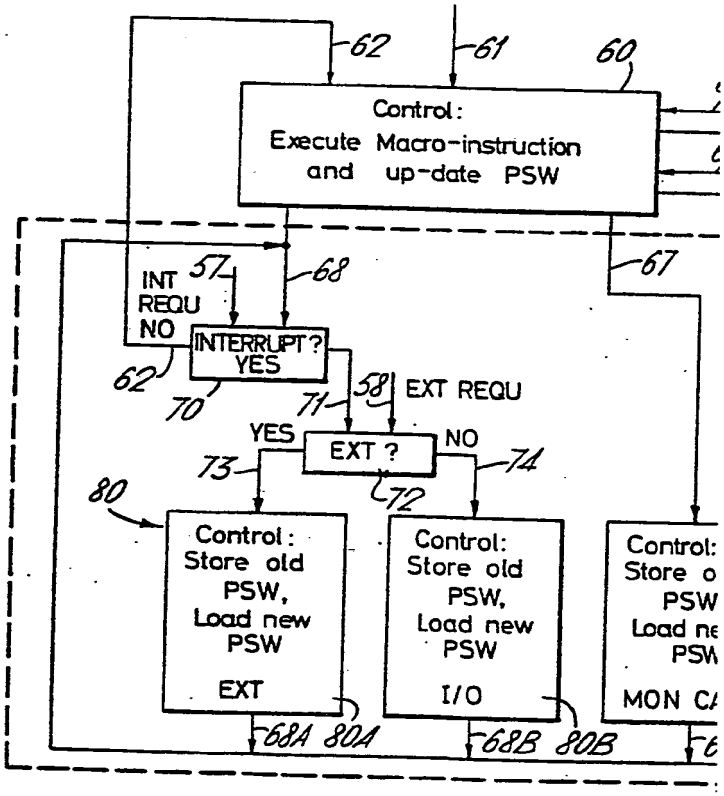


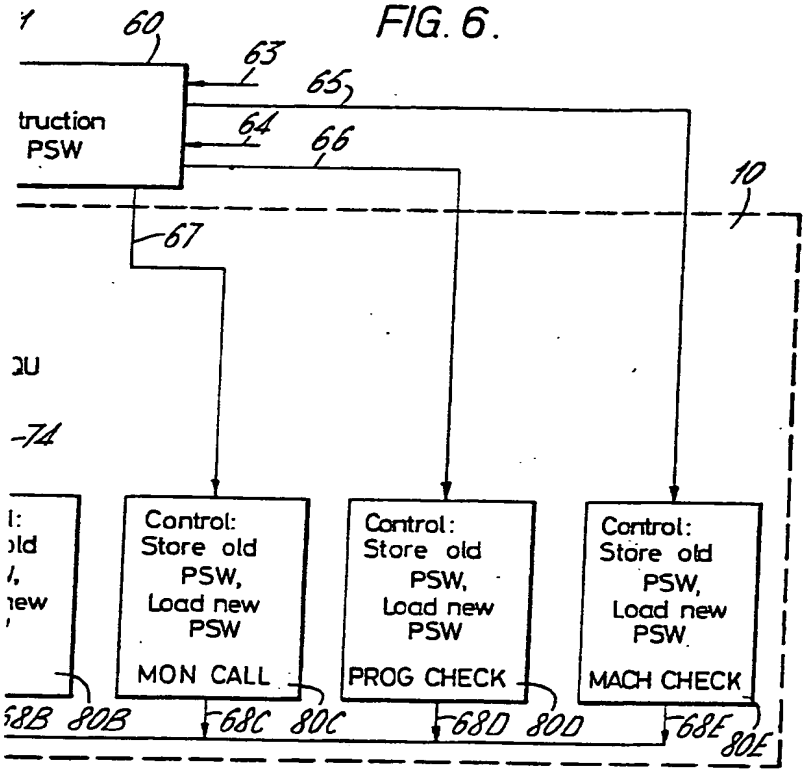
FIG. 5.




62
61
60
Control:
Execute Macro-instruction
and up-date PSW
67
57
68
INT
REQU
NO
INTERRUPT?
YES
62
70
71
58
EXT REQU
YES
EXT ?
NO
73
74
72
80
Control:
Store old
PSW,
Load new
PSW
EXT
Control:
Store old
PSW,
Load new
PSW
I/O
Control:
Store o
PSW
Load ne
PSW
MON C
68A
80A
68B
80B

1108801 COMPLETE SPECIFICATION
5 SHEETS *This drawing is a reproduction of the Original on a reduced scale*
Sheet 5

FIG. 6.



1108801 COMPLETE SPECIFICATION
5 SHEETS This drawing is a reproduction of the Original on a reduced scale
Sheet 5

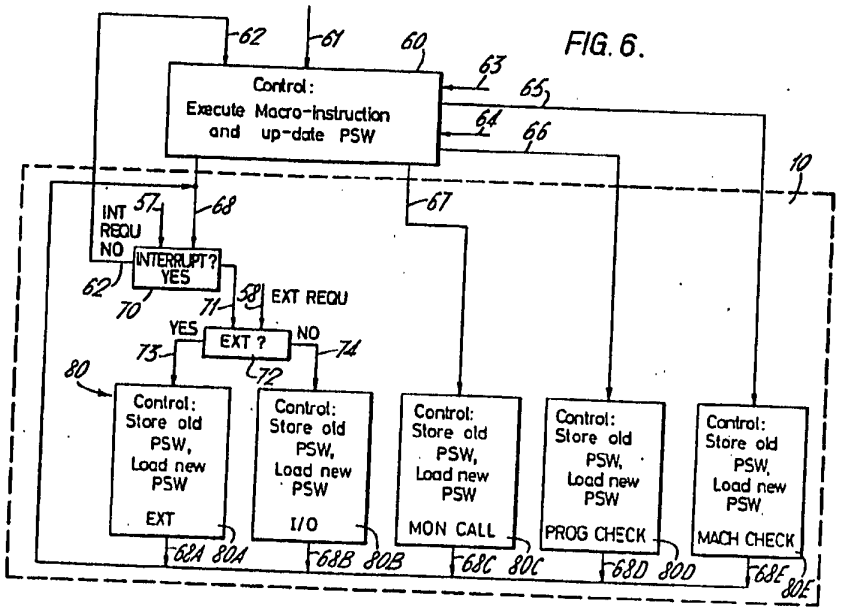


FIG. 6.